Weighted Model Counting with Conditional Weights

Abstract

AAAI creates proceedings, working notes, and technical reports directly from electronic source furnished by the authors. To ensure that all papers in the publication have a uniform appearance, authors must adhere to the following instructions.

1 Introduction

Introduce WMC (Sang, Beame, and Kautz 2005)

Over the last fifteen years, WMC has been extended and generalised in many ways, e.g., to handle continuous probability distributions (Belle, Passerini, and Van den Broeck 2015), first-order probabilistic theories (Van den Broeck et al. 2011; Gogate and Domingos 2016), and infinite domains (Belle 2017). Furthermore, WMC has been generalised from weights in the interval $[0, 1]$ or positive real numbers to an arbitrary semiring, making it applicable to domains such as parsing, dynamic programming, constraint programming, databases, networks, etc. (Kimmig, Van den Broeck, and De Raedt 2017). It has been applied to Bayesian networks (Sang, Beame, and Kautz 2005; Darwiche 2009) and probabilistic graphical models more generally (Choi, Kisa, and Darwiche 2013), probabilistic logic programming languages (Fierens et al. 2015) and probabilistic programs with discrete probability distributions (Holtzen, Van den Broeck, and Millstein 2020). Many WMC algorithms have been proposed based primarily either on knowledge compilation (Darwiche 2004; Oztok and Darwiche 2015; Lagniez and Marquis 2017) or search (Sang, Beame, and Kautz 2005).

The most well-studied version of WMC assigns weights to models based on weights on literals, i.e., the weight of a model is the product of the weights of all literals in it. This severely restricts what probability distributions can be represented, and this limitation is usually overcome by increasing the size of the problem, i.e., adding more literals.

We show that while this workaround is always possible, it can be significantly more efficient to consider *conditional weights* on literals. Similarly to conditional probabilities, conditional weights allow us to represent probabilities in a manner that is both efficient and interpretable. Not having additional variables that are only needed to manipulate the representation into a specific format helps with both efficiency and interpretability. Similarly to Bayesian networks, the size of the representation scales with independence and literal-weight WMC is a special case when everything is independent.

We demonstrate a specific example of this by presenting a new way to encode Bayesian networks into instances of WMC and adapting a WMC algorithm ADDMC to run on the new format. ADDMC (Dudek, Phan, and Vardi 2020) is a recently-proposed algorithm for WMC based on manipulating functions on Boolean algebras using an efficient representation for such functions known as *algebraic decision diagrams* (ADDs) (Bahar et al. 1997). We show that with a few minor modifications the algorithm can be adapted to work on our new WMC format

Showing that the new encoding produces an asymptotically smaller problem (w.r.t. the numbers of literals and ADDs) and results in faster inferences.

2 Related Work

WMC for Bayesian networks. Hitherto, four techniques have been proposed for encoding Bayesian networks into instances of WMC. We will identify them based on the initials of authors as well as publications years: `d02` (Darwiche 2002), `sbk05` (Sang, Beame, and Kautz 2005), `cd05` (Chavira and Darwiche 2005), and `cd06` (Chavira and Darwiche 2006). Below we summarise the observed performance differences among them. Sang, Beame, and Kautz (2005) claim that `sbk05` is a smaller encoding than `d02` with respect to both the number of clauses and the number of variables but provide no experimental comparison. Chavira and Darwiche (2005) compare `cd05` with `d02` by measuring the time it takes to compile either encoding into an arithmetic circuit (but do not measure inference time). The more recent encoding `cd05` always compiles faster and results in a smaller arithmetic circuit (as measured by the number of edges). In their subsequent paper, the same authors perform two sets of experiments (that are relevant to this summary) (Chavira and Darwiche 2006). First, they compile `cd05` and `cd06` encodings into d-DNNF (i.e., deterministic decomposable negation normal form (Darwiche 2001)), measuring both compilation time and numbers of edges in the d-DNNF diagram. The results are mostly in favour of `cd06`. Second, they compare the inference time of `sbk05` run with Ca-

chet (Sang et al. 2004) with the compile times of `cd05` and `cd06`, but only on five (types of) instances. In these experiments, `cd06` is always faster than `cd05`, while the comparison with `sbk05` is mixed. Sometimes `cd06` is orders of magnitude faster than `sbk05`, sometimes slightly slower. The performance difference between `sbk05` and `cd05` is even harder to judge: `sbk05` is better on three out of five instances and worse on the remaining two. Based on this description, one would expect `cd06` to be faster than both `cd05` and `sbk05`, both of which should be faster than `d02`. The experiments in Section 6, however, strongly disagree with this prediction, showing that the quality of an encoding depends strongly on the underlying search algorithm or compilation technique.

ADDs and their use in probabilistic inference. ADDs provide an efficient way to manipulate functions from a Boolean algebra to any algebraic structure (most commonly the real numbers) (Bahar et al. 1997). They have been used to represent value functions in Markov decision processes (Hoey et al. 1999) and, for Bayesian network inference, to represent each conditional probability table (CPT) as an ADD (Zhao, Melibari, and Poupart 2015) as well as by combining ADDs and arithmetic circuits into a single representation (Chavira and Darwiche 2007). ADDs are particularly advantageous in this situation because of their ability to fully exploit context-specific independence, i.e., observable structure within a CPT that is not inherited from the structure of the Bayesian network (Boutilier et al. 1996).

3 Boolean Algebras, Power Sets, and Propositional Logic

Let $\mathcal{L}$ be a propositional logic with atoms a and b, and let $U = \{a, b\}$. Then 2^U, the power set of U, is the set of all models of $\mathcal{L}$, and 2^{2^U} is the set of all formulas. These sets can also be represented as Boolean algebras (e.g., $(2^{2^U}, \wedge, \vee, \neg, \bot, \top)$) with a partial order $\leq$ that corresponds to set inclusion $\subseteq$—see Table 1 for examples of how various elements can be represented in both notations. Most importantly, note that the word *atom* has completely different meanings in logic and in Boolean algebras. An atom in $\mathcal{L}$ is an atomic formula, i.e., an element of U, whereas an atom in a Boolean algebra is (in set-theoretic terms) a singleton set. For instance, an atom in 2^{2^U} corresponds to a model of $\mathcal{L}$, i.e., an element of 2^U. Unless referring specifically to a logic, we will use the algebraic definition of an atom while referring to logical atoms as *variables*. In the rest of the paper, for any set U, we will use set-theoretic notation for 2^U and Boolean-algebraic notation for 2^{2^U}, except for (Boolean) atoms in 2^{2^U} that are denoted as $\{x\}$ for some model $x \in 2^U$.

3.1 The Space of Functions on Boolean Algebras

We build on the definitions of multiplication and projection by Dudek, Phan, and Vardi (2020) and define more operations that can be used to manipulate functions from Boolean algebras to non-negative real numbers. All of these operations have efficient implementations in the CUDD (Somenzi 2015) package for manipulating ADDs (among other things) that is used by ADDMC (Dudek, Phan, and Vardi 2020).

Definition 1 (Operations on functions)**.** Let $\alpha\colon 2^X \to \mathbb{R}_{\geq 0}$ and $\beta\colon 2^Y \to \mathbb{R}_{\geq 0}$ be functions, $p \in \mathbb{R}_{\geq 0}$, and $x \in X$. We define the following operations:

Addition: $\alpha + \beta\colon 2^{X \cup Y} \to \mathbb{R}_{\geq 0}$ is such that $(\alpha + \beta)(T) = \alpha(T \cap X) + \beta(T \cap Y)$ for all $T \in 2^{X \cup Y}$.

Multiplication: $\alpha \cdot \beta\colon 2^{X \cup Y} \to \mathbb{R}_{\geq 0}$ is such that $(\alpha \cdot \beta)(T) = \alpha(T \cap X) \cdot \beta(T \cap Y)$ for all $T \in 2^{X \cup Y}$.

Scalar multiplication: $p\alpha\colon 2^X \to \mathbb{R}_{\geq 0}$ is such that $(p\alpha)(T) = p \cdot \alpha(T)$ for all $T \in 2^X$.

Complement: $\overline{\alpha}\colon 2^X \to \mathbb{R}_{\geq 0}$ is such that $\overline{\alpha}(T) = 1 - \alpha(T)$ for all $T \in 2^X$.

Projection: $\exists_x \alpha\colon 2^{X \setminus \{x\}} \to \mathbb{R}_{\geq 0}$ is such that $(\exists_x \alpha)(T) = \alpha(T) + \alpha(T \cup \{x\})$ for all $T \in 2^{X \setminus \{x\}}$. For any $Z = \{z_1, \ldots, z_n\} \subseteq X$, we write $\exists_Z$ to mean $\exists_{z_1} \ldots \exists_{z_n}$.

Observation 1. *Let U be a set, and $\mathcal{V} = \{\alpha\colon 2^X \to \mathbb{R}_{\geq 0} \mid X \subseteq U\}$. Then $\mathcal{V}$ is a semi-vector space with three additional operations: (non-scalar) multiplication, complement, and projection. Specifically, note that both addition and multiplication are both associative and commutative.*

We end the discussion on function spaces by defining several special functions: unit $1\colon 2^\emptyset \to \mathbb{R}_{\geq 0}$ defined as $1(\emptyset) = 1$, zero $0\colon 2^\emptyset \to \mathbb{R}_{\geq 0}$ defined as $0(\emptyset) = 0$, and function $[a]\colon 2^{\{a\}} \to \mathbb{R}_{\geq 0}$ defined as $[a](\emptyset) = 0$, $[a](\{a\}) = 1$ for any a. Henceforth, for any function $\alpha\colon 2^X \to \mathbb{R}_{\geq 0}$ and any set T, we will write $\alpha(T)$ to mean $\alpha(T \cap X)$.

4 WMC as a Measure

Let U be a set. A *measure* is a function $\mu\colon 2^{2^U} \to \mathbb{R}_{\geq 0}$ such that $\mu(\bot) = 0$, and $\mu(a \vee b) = \mu(a) + \mu(b)$ for all $a, b \in 2^{2^U}$ whenever $a \wedge b = \bot$. A *weight function* is a function $\nu\colon 2^U \to \mathbb{R}_{\geq 0}$. A weight function is *factored* if $\nu = \prod_{x \in U} \nu_x$ for some functions $\nu_x\colon 2^{\{x\}} \to \mathbb{R}_{\geq 0}$, $x \in U$. We say that a weight function $\nu\colon 2^U \to \mathbb{R}_{\geq 0}$ *induces* a measure $\mu_\nu\colon 2^{2^U} \to \mathbb{R}_{\geq 0}$ if $\mu_\nu(x) = \sum_{\{u\} \leq x} \nu(u)$.

Theorem 1. *The function μ_ν is a measure.*

Finally, a measure $\mu\colon 2^{2^U} \to \mathbb{R}_{\geq 0}$ is *factorable* if there exists a factored weight function $\nu\colon 2^U \to \mathbb{R}_{\geq 0}$ that induces μ. In this formulation, the process of calculating the value of $\mu_\nu(x)$ for some $x \in 2^{2^U}$ with a given definition of ν is known as WMC.

Relation to the classical (logic-based) view of WMC. Let $\mathcal{L}$ be a propositional logic with two atoms a and b as in Section 3 and $w\colon \{a, b, \neg a, \neg b\} \to \mathbb{R}_{\geq 0}$ a *weight function* defined as $w(a) = 0.3$, $w(\neg a) = 0.7$, $w(b) = 0.2$, $w(\neg b) = 0.8$. Furthermore, let Δ be a theory in $\mathcal{L}$ with a sole axiom a. Then Δ has two models: $\{a, b\}$ and $\{a, \neg b\}$

Table 1: Notation for a logic with two atoms. The elements in both columns are listed in the same order.

Name in logic	Boolean-algebraic notation	Set-theoretic notation
Atoms (elements of U)	a, b	a, b
Models (elements of 2^U)	$\neg a \wedge \neg b, a \wedge \neg b, \neg a \wedge b, a \wedge b$	$\emptyset, \{a\}, \{b\}, \{a, b\}$
Formulas (elements of 2^{2^U})	$\top$	$\{\emptyset, \{a\}, \{b\}, \{a, b\}\}$
	$\neg a \vee \neg b, a \to b$	$\{\emptyset, \{a\}, \{b\}\}, \{\emptyset, \{a\}, \{a, b\}\}$
	$b \to a, a \vee b$	$\{\emptyset, \{b\}, \{a, b\}\}, \{\{a\}, \{b\}, \{a, b\}\}$
	$\neg b, \neg a, a \leftrightarrow b$	$\{\emptyset, \{a\}\}, \{\emptyset, \{b\}\}, \{\emptyset, \{a, b\}\}$
	$(a \wedge \neg b) \vee (b \wedge \neg a), a, b$	$\{\{a\}, \{b\}\}, \{\{a\}, \{a, b\}\}, \{\{b\}, \{a, b\}\}$
	$\neg a \wedge \neg b, a \wedge \neg b, \neg a \wedge b, a \wedge b$	$\{\emptyset\}, \{\{a\}\}, \{\{b\}\}, \{\{a, b\}\}$
	$\bot$	$\emptyset$

and its WMC (Chavira and Darwiche 2008) is

$$\begin{aligned} \mathrm{WMC}(\Delta) &= \sum_{\omega \models \Delta} \prod_{\omega \models l} w(l) \\ &= w(a)w(b) + w(a)w(\neg b) = 0.3. \end{aligned} \tag{1}$$

Alternatively, we can define $\nu_a \colon 2^{\{a\}} \to \mathbb{R}_{\geq 0}$ as $\nu_a(\{a\}) = 0.3$, $\nu_a(\emptyset) = 0.7$ and $\nu_b \colon 2^{\{b\}} \to \mathbb{R}_{\geq 0}$ as $\nu_b(\{b\}) = 0.2$, $\nu_b(\emptyset) = 0.8$. Let μ be the measure on 2^{2^U} induced by $\nu = \nu_a \cdot \nu_b$. Then, equivalently to Eq. (1), we can write

$$\begin{aligned} \mu(a) &= \nu(\{a, b\}) + \nu(\{a\}) \\ &= \nu_a(\{a\})\nu_b(\{b\}) + \nu_a(\{a\})\nu_b(\emptyset) = 0.3. \end{aligned}$$

4.1 Not All Measures Are Factorable

WMC with weights defined on literals is only able to capture a subset of all possible measures on the Boolean algebra. This can be illustrated with a simple example.

Example 1. Let $U = \{a, b\}$ be a set of atoms and $\mu \colon 2^{2^U} \to \mathbb{R}_{\geq 0}$ a measure defined as:[1]

$$\begin{aligned} \mu(a \wedge b) &= 0.72, \\ \mu(a \wedge \neg b) &= 0.18, \\ \mu(\neg a \wedge b) &= 0.07, \\ \mu(\neg a \wedge \neg b) &= 0.03. \end{aligned}$$

If μ could be represented using traditional (factored) WMC, we would have to find two weight functions $\nu_a \colon 2^{\{a\}} \to \mathbb{R}_{\geq 0}$ and $\nu_b \colon 2^{\{b\}} \to \mathbb{R}_{\geq 0}$ such that $\nu = \nu_a \cdot \nu_b$ induces μ, i.e., ν_a and ν_b would have to satisfy this system of equations:

$$\begin{aligned} \nu_a(\{a\}) \cdot \nu_b(\{b\}) &= 0.72 \\ \nu_a(\{a\}) \cdot \nu_b(\emptyset) &= 0.18 \\ \nu_a(\emptyset) \cdot \nu_b(\{b\}) &= 0.07 \\ \nu_a(\emptyset) \cdot \nu_b(\emptyset) &= 0.03, \end{aligned}$$

which has no solutions.

Alternatively, we can let b depend on a and consider weight functions $\nu_a \colon 2^{\{a\}} \to \mathbb{R}_{\geq 0}$ and $\nu_b \colon 2^{\{a,b\}} \to \mathbb{R}_{\geq 0}$ defined as $\nu_a(\{a\}) = 0.9$, $\nu_a(\emptyset) = 0.1$, and $\nu_b(\{a, b\}) = 0.8$, $\nu_b(\{a\}) = 0.2$, $\nu_b(\{b\}) = 0.7$, $\nu_b(\emptyset) = 0.3$. One can easily check that with these definitions ν indeed induces μ.

[1] The value of μ on any other element of the Boolean algebra can be deduced using the definition.

Note that in this case we chose to interpret ν_b as $\Pr(b \mid a)$ while—with a different definition of ν_b that represents the joint probability distribution $\Pr(a, b)$—ν_b by itself could induce μ. In general, however, factorising the full weight function into several smaller functions often results in weight functions with smaller domains which leads to increased efficiency and decreased memory usage (Dudek, Phan, and Vardi 2020).

Since many measures of interests may not be factorable, a well-known way to encode them into instances of WMC is by adding more literals (Chavira and Darwiche 2008). We can indeed show that this is always possible, however, as ensuing sections will demonstrate, it often makes the inference task much harder.

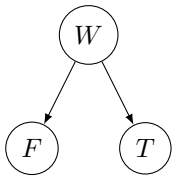
Theorem 2. *For any set U and measure $\mu \colon 2^{2^U} \to \mathbb{R}_{\geq 0}$, there exists a set $V \supseteq U$, a factorable measure $\mu' \colon 2^{2^V} \to \mathbb{R}_{\geq 0}$, and a formula $f \in 2^{2^V}$ such that $\mu(x) = \mu'(x \wedge f)$ for all formulas $x \in 2^{2^U}$.*

5 Encoding Bayesian Networks Using Conditional Weights

In this section, we describe a way to encode Bayesian networks into WMC without restricting oneself to factorable measures and thus having to add extra variables to maintain literal independence. In line with the names of previous encodings, we shall call this encoding `db21`.

A Bayesian network is a directed acyclic graph with random variables as vertices that defines a probability distribution over them. Let $\mathcal{V}$ denote this set of random variables. For any random variable $X \in \mathcal{V}$, let $\operatorname{im} X$ denote its set of values and $\mathrm{pa}(X)$ its set of parents. The full probability distribution is then equal to $\prod_{X \in \mathcal{V}} \Pr(X \mid \mathrm{pa}(X))$. For discrete Bayesian networks (and we only consider discrete networks in this paper), each factor of this product can be represented by a CPT. See Fig. 1 for an example Bayesian network which we will refer to throughout this section. For this network, $\mathcal{V} = \{W, F, T\}$, $\mathrm{pa}(W) = \emptyset$, $\mathrm{pa}(F) = \mathrm{pa}(T) = \{W\}$, $\operatorname{im} W = \operatorname{im} F = \{0, 1\}$, and $\operatorname{im} T = \{l, m, h\}$.

Definition 2 (Indicator variables)**.** Let $X \in \mathcal{V}$ be a random variable. If X is binary (i.e., $|\operatorname{im} X| = 2$), we can arbitrary identify one of the values as 1 and the other one as 0 (i.e,



w	$\Pr(W = w)$
1	0.5
0	0.5

w	f	$\Pr(F = f \mid W = w)$
1	1	0.6
1	0	0.4
0	1	0.1
0	0	0.9

w	t	$\Pr(T = t \mid W = w)$
1	l	0.2
1	m	0.4
1	h	0.4
0	l	0.6
0	m	0.3
0	h	0.1

Figure 1: An example Bayesian network with its CPTs

$\operatorname{im} X \cong \{0, 1\}$). Then X can be represented by a single *indicator variable* $\lambda_{X=1}$. For notational simplicity, for any set S, we write $\lambda_{X=0} \in S$ or $S = \{\lambda_{X=0}, \dots\}$ to mean $\lambda_{X=1} \notin S$.

On the other hand, if X is not binary, we represent X with $|\operatorname{im} X|$ indicator variables, one for each value. We let

$$\mathcal{E}(X) = \begin{cases} \{\lambda_{X=1}\} & \text{if } |\operatorname{im} X| = 2 \\ \{\lambda_{X=x} \mid x \in \operatorname{im} X\} & \text{otherwise.} \end{cases}$$

denote the set of indicator variables for X and $\mathcal{E}^*(X) = \mathcal{E}(X) \cup \bigcup_{Y \in \operatorname{pa}(X)} \mathcal{E}(Y)$ denote the set of indicator variables for X and its parents in the Bayesian network. Finally, let $U = \bigcup_{X \in \mathcal{V}} \mathcal{E}(X)$ denote the set of all indicator variables for all random variables in the Bayesian network.

For example, in the Bayesian network from Fig. 1, $\mathcal{E}^*(T) = \{\lambda_{T=l}, \lambda_{T=m}, \lambda_{T=h}, \lambda_{W=1}\}$.

Algorithm 1 shows how a Bayesian network with vertices $\mathcal{V}$ can be represented as a weight function $\phi\colon 2^U \to \mathbb{R}_{\geq 0}$. The algorithm begins with the unit function and multiplies it by $\mathrm{CPT}_X\colon 2^{\mathcal{E}^*(X)} \to \mathbb{R}_{\geq 0}$ for each random variable $X \in \mathcal{V}$. We call each such function a *conditional weight function* as it represents a conditional probability distribution. However, the distinction is primarily a semantic one: a function $2^{\{a,b\}} \to \mathbb{R}_{\geq 0}$ can represent $\Pr(a \mid b)$, $\Pr(b \mid a)$, or something else entirely.

For a binary random variable X, CPT_X is simply a sum of smaller functions, one for each row of the CPT. If X has more than two values, we also multiply CPT_X by some 'clause' functions that restrict the value of $\phi(T)$ to zero whenever $|\mathcal{E}(X) \cap T| \neq 1$. For the Bayesian network in Fig. 1, we get:

$$\begin{aligned} \mathrm{CPT}_\mathrm{F} &= 0.6[\lambda_{F=1}] \cdot [\lambda_{W=1}] + 0.4[\lambda_{F=0}] \cdot [\lambda_{W=1}] \\ &\quad + 0.1[\lambda_{F=1}] \cdot [\lambda_{W=0}] + 0.9[\lambda_{F=0}] \cdot [\lambda_{W=0}], \\ \mathrm{CPT}_\mathrm{T} &= ([\lambda_{T=l}] + [\lambda_{T=m}] + [\lambda_{T=h}]) \\ &\quad \cdot (\overline{[\lambda_{T=l}]} + \overline{[\lambda_{T=m}]}) \cdot (\overline{[\lambda_{T=l}]} + \overline{[\lambda_{T=h}]}) \\ &\quad \cdot (\overline{[\lambda_{T=m}]} + \overline{[\lambda_{T=h}]}) \cdot \dots . \end{aligned}$$

5.1 Correctness

Algorithm 1 produces a function with a Boolean algebra as its domain. This function can be represented by an ADD. The core of ADDMC works by taking an ADD $\psi\colon 2^U \to \mathbb{R}_{\geq 0}$ (expressed as a product of smaller ADDs) and returning $(\exists_U \psi)(\emptyset)$ (Dudek, Phan, and Vardi 2020). In this section, we prove that the function ϕ produced by Algorithm 1

Algorithm 1: Encoding a Bayesian network

Data: vertices $\mathcal{V}$, probability distribution $\Pr$
Result: $\phi\colon 2^U \to \mathbb{R}_{\geq 0}$
$\phi \leftarrow 1$;
for $X \in \mathcal{V}$ **do**
  let $\operatorname{pa}(X) = \{Y_1, \dots, Y_n\}$;
  $\mathrm{CPT}_X \leftarrow 0$;
  if $|\operatorname{im} X| = 2$ **then**
    for $(y_i)_{i=1}^n \in \prod_{i=1}^n \operatorname{im} Y_i$ **do**
      $p_1 \leftarrow \Pr(X = 1 \mid y_1, \dots, y_n)$;
      $p_0 \leftarrow \Pr(X \neq 1 \mid y_1, \dots, y_n)$;
      $\mathrm{CPT}_X \leftarrow \mathrm{CPT}_X$
        $+ p_1[\lambda_{X=1}] \cdot \prod_{i=1}^n [\lambda_{Y_i=y_i}]$
        $+ p_0\overline{[\lambda_{X=1}]} \cdot \prod_{i=1}^n [\lambda_{Y_i=y_i}]$;
  else
    let $\operatorname{im} X = \{x_1, \dots, x_m\}$;
    for $x \in \operatorname{im} X$ **and** $(y_i)_{i=1}^n \in \prod_{i=1}^n \operatorname{im} Y_i$ **do**
      $p_x \leftarrow \Pr(X = x \mid y_1, \dots, y_n)$;
      $\mathrm{CPT}_X \leftarrow \mathrm{CPT}_X$
        $+ p_x[\lambda_{X=x}] \cdot \prod_{i=1}^n [\lambda_{Y_i=y_i}]$
        $+ \overline{[\lambda_{X=x}]} \cdot \prod_{i=1}^n [\lambda_{Y_i=y_i}]$;
    $\mathrm{CPT}_X \leftarrow \mathrm{CPT}_X \cdot (\sum_{i=1}^m [\lambda_{X=x_i}])$
      $\cdot \prod_{i=1}^m \prod_{j=i+1}^m (\overline{[\lambda_{X=x_i}]} + \overline{[\lambda_{X=x_j}]})$;
  $\phi \leftarrow \phi \cdot \mathrm{CPT}_X$;
return ϕ;

can be used by ADDMC to correctly compute any marginal probability of the Bayesian network that was encoded as ϕ.[2] First, Lemma 1 shows that any conditional weight function produces the right answer when given an appropriate encoding of variable-value assignments for a conditional probability in the Bayesian network. Then, Lemma 2 shows that ϕ represents the full probability distribution of the Bayesian network, i.e., it gives the right probabilities for the right inputs and zero otherwise. We end with Theorem 3 that shows how ϕ can be combined with an encoding of a single variable-value assignment so that ADDMC would compute its marginal probability.

Lemma 1. *Let $X \in \mathcal{V}$ be a random variable with parents $\mathrm{pa}(X) = \{Y_1, \ldots, Y_n\}$. Then $\mathrm{CPT}_X \colon 2^{\mathcal{E}^*(X)} \to \mathbb{R}_{\geq 0}$ is such that for any $x \in \operatorname{im} X$ and $(y_1, \ldots, y_n) \in \prod_{i=1}^n \operatorname{im} Y_i$,*

$$\mathrm{CPT}_X(T) = \Pr(X = x \mid Y_1 = y_1, \ldots, Y_n = y_n),$$

where $T = \{\lambda_{X=x}\} \cup \{\lambda_{Y_i=y_i} \mid i = 1, \ldots, n\}$.

Lemma 2. *Let $\mathcal{V} = \{X_1, \ldots, X_n\}$. Then*

$$\phi(T) = \begin{cases} \Pr(x_1, \ldots, x_n) & \text{if } T = \{\lambda_{X_i=x_i}\}_{i=1}^n \text{ for} \\ & \text{some } (x_i)_{i=1}^n \in \prod_{i=1}^n \operatorname{im} X_i \\ 0 & \text{otherwise,} \end{cases}$$

for all $T \in 2^U$.

Theorem 3. *For any $X \in \mathcal{V}$ and $x \in \operatorname{im} X$,*

$$(\exists_U(\phi \cdot [\lambda_{X=x}]))(\emptyset) = \Pr(X = x).$$

5.2 Textual Representation

Algorithm 1 encodes a Bayesian network into a function on a Boolean algebra, but how does it relate to the standard interpretation of a WMC encoding as a formula in conjunctive normal form (CNF) together with a collection of weights? The factors of ϕ that restrict the values of indicator variables for non-binary random variables are already expressed as a product of sums of 0/1-valued functions, i.e., a kind of CNF. Disregarding these functions, each conditional weight function CPT_X is represented by a sum with a term for every subset of $\mathcal{E}^*(X)$. To encode these terms, we introduce *extended weight clauses* to the WMC format used by Cachet (Sang et al. 2004). For instance, here is a representation of the Bayesian network from Fig. 1:

$$\begin{array}{llllll}
\lambda_{T=l} & \lambda_{T=m} & \lambda_{T=h} & & 0 \\
 & -\lambda_{T=l} & -\lambda_{T=m} & & 0 \\
 & -\lambda_{T=l} & -\lambda_{T=h} & & 0 \\
 & -\lambda_{T=m} & -\lambda_{T=h} & & 0 \\
w & \lambda_{W=1} & & 0.5 & 0.5 \\
w & \lambda_{F=1} & \lambda_{W=1} & 0.6 & 0.4 \\
w & \lambda_{F=1} & -\lambda_{W=1} & 0.1 & 0.9 \\
w & \lambda_{T=l} & \lambda_{W=1} & 0.2 & 1 \\
w & \lambda_{T=m} & \lambda_{W=1} & 0.4 & 1 \\
w & \lambda_{T=h} & \lambda_{W=1} & 0.4 & 1 \\
w & \lambda_{T=l} & -\lambda_{W=1} & 0.6 & 1 \\
w & \lambda_{T=m} & -\lambda_{W=1} & 0.3 & 1 \\
w & \lambda_{T=h} & -\lambda_{W=1} & 0.1 & 1
\end{array}$$

[2] It can just as well compute any probability expressed using the random variables in $\mathcal{V}$, but we focus mostly on marginal probabilities in our experiments.

where each indicator variable is eventually replaced with a unique positive integer. Each line prefixed with a w can be split into four parts: the 'main' variable (always not negated), conditions (possibly none), and two weights. For example, the line

$$w \quad \lambda_{T=m} \quad -\lambda_{W=1} \quad 0.3 \quad 1$$

encodes the function $0.3[\lambda_{T=m}] \cdot \overline{[\lambda_{W=1}]} + 1\overline{[\lambda_{T=m}]} \cdot \overline{[\lambda_{W=1}]}$ and can be interpreted as defining two conditional weights: $\nu(T = m \mid W = 0) = 0.3$, and $\nu(T \neq m \mid W = 0) = 1$, the former of which corresponds to a row in the CPT of T while the latter is artificially added as part of the encoding. In our encoding of Bayesian networks, it is always the case that, in each weight clause, either both weights sum to one, or the second weight is equal to one. Finally, note that (without any additional restrictions), the measure induced by these weight functions is not probabilistic (i.e., $\mu(\top)$ may not be equal to one).

5.3 Changes to ADDMC

ADDMC constructs the *Gaifman graph* (Gaifman 1982) of the input CNF formula as an aid for the algorithm's heuristics. This graph has as vertices the variables of the formula, and there is an edge between two variables u and v if there is a clause in the formula that contains both u and v. We extend this definition to functions on Boolean algebras, i.e., the factors of ϕ. For any pair of distinct variables $u, v \in U$, we draw an edge between them in the Gaifman graph if there is a function $\alpha \colon 2^X \to \mathbb{R}_{\geq 0}$ that is a factor of ϕ such that $u \in X$ and $v \in X$. For instance, a factor such as CPT_X will enable edges between all distinct pairs of variables in $\mathcal{E}^*(X)$.

Even though the function ϕ produced by Algorithm 1 is constructed to have 2^U as its domain, sometimes the domain is effectively reduced to 2^V for some $V \subset U$ by the ADD manipulation algorithms that optimise the ADD representation of a function. For a simple example, consider $\alpha : 2^{\{a\}} \to \mathbb{R}_{\geq 0}$ defined as $\alpha(\{a\}) = \alpha(\emptyset) = 0.5$. Then α can be reduced to $\alpha' \colon 2^{\emptyset} \to \mathbb{R}_{\geq 0}$ defined as $\alpha'(\emptyset) = 0.5$. To compensate for these reductions, for the original WMC format with a weight function $w \colon U \cup \{\neg u \mid u \in U\} \to \mathbb{R}_{\geq 0}$, ADDMC would multiply its computed answer by $\prod_{u \in U \setminus V} w(u) + w(\neg u)$. With the new WMC format, we instead multiply the answer by $2^{|U \setminus V|}$. Each 'excluded' variable $u \in U \setminus V$ satisfies two properties: all weights associated with u are equal to 0.5 (otherwise the corresponding CPT would depend on u, and u would not be excluded), and all other CPTs are independent of u (or they may have a trivial dependence, where the probability stays the same if u is replaced with its complement). Thus, the CPT that corresponds to u still multiplies every model by 0.5, but the number of models being considered by the ADDMC is halved. To correct for this, we multiply the final answer by two for every $u \in U \setminus V$.

6 Experimental Comparison

In this section, we describe an experimental study comparing the five WMC encodings for Bayesian networks when

run with ADDMC. The experiments were run on Intel Xeon Gold 6138 processor with an 8 GB memory limit.

For each Bayesian network, we need to choose a probability to compute. Whenever a Bayesian network comes with an evidence file, we compute the probability of evidence. Otherwise, let X denote the last-mentioned vertex in the Bayesian network. If true is a valid value of X, we compute the marginal probability of $X = \mathsf{true}$. Otherwise, we pick the value of X which is listed first and calculate its marginal probability.

For all encodings other than `db21`, we use their implementation in Ace 3.0[3] with `-encodeOnly` and `-noEclause` flags. However, Ace was not used to encode evidence, as preliminary experiments revealed that the evidence-encoding implementation contains bugs that can lead to incorrect answers or a Java exception being thrown on some instances of the data set (and the source code is not publicly available). Instead, we simply list all the evidence as additional clauses in the encoding.

Note that both `cd05` and `cd06` purposefully produce overly relaxed encodings that contain extra models and thus yield incorrect probabilities (Chavira and Darwiche 2005, 2006). These additional models are supposed to be filtered out during circuit compilation (Chavira and Darwiche 2005), but this is not easily achievable with ADDMC. Nonetheless, we include both encodings in the experiments.

While encoding time itself was not measured because of different languages of implementation, it is worth noting that `db21` is linear in the total number of CPT rows and so is unlikely to be slower than, e.g., `cd06`, which relies on solving NP-complete problems as part of the encoding process (Chavira and Darwiche 2006).

Data. For experimental data, we use the Bayesian networks available with Ace and Cachet[4], most of which happen to be binary. We classify them into the following seven categories: • DQMR and • Grid networks as described by Sang, Beame, and Kautz (2005), • Friends and Smokers, • Mastermind, and • Random Blocks from the work of Chavira, Darwiche, and Jaeger (2006), • remaining binary Bayesian networks that include Plan Recognition (Sang, Beame, and Kautz 2005), Students and Professors (Chavira, Darwiche, and Jaeger 2006), and `tcc4f`, and • non-binary classic Bayesian networks (`alarm`, `diabetes`, `hailfinder`, `mildew`, `munin1–4`, `pathfinder`, `pigs`, `water`).

Observations.

- The order of the encodings from best to worst is exactly the opposite of that in the previous literature.
- Our encoding is particularly promising on instances of Grid networks. They are set up so that the entire Bayesian network is relevant to the query.
- However, `db21` struggles with instances from Mastermind and Random Blocks domains (but so does `sbk05`). We conjecture that this is so either because the particular structure of these problems confuse the heuristics used by ADDMC or because these instances allow for significant simplifications that are exploited by `d02` but not `db21`.
- An observation from the cumulative plot: `db21` solves the same number of instances in 6.374 s as `d02` does in 1000 s.

[3]http://reasoning.cs.ucla.edu/ace/

[4]https://www.cs.rochester.edu/u/kautz/Cachet/

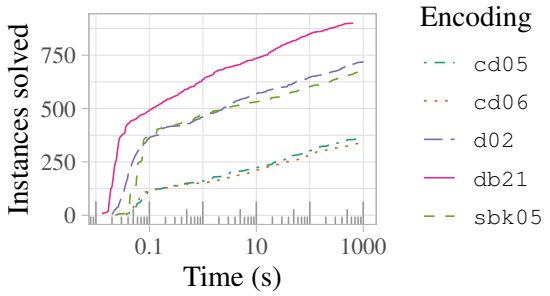


Figure 2: Cumulative number of instances solved by ADDMC over time using each encoding

Table 2: The numbers of instances solved by ADDMC with each encoding (uniquely, faster than with others, and in total) under the described time and memory constraints (out of 1216 instances)

Encoding	Unique	Fastest	Total
`cd05`	2	3	372
`cd06`	0	1	351
`d02`	41	99	726
`db21`	228	871	901
`sbk05`	6	36	687

Explaining the performance benefits. Let $n = |\mathcal{V}|$ be the number of vertices in the Bayesian network, $d = \max_{X \in \mathcal{V}} |\mathrm{pa}(X)|$ the maximum in-degree (i.e., number of parents), and $v = \max_{X \in \mathcal{V}} |\operatorname{im} X|$ the maximum number of values per variable. Table 3 shows how `db21` has both fewer variables and fewer ADDs than any other encoding. However, note that these are upper bounds and most encodings (including `db21`) can be smaller in certain situations (e.g., with binary random variables or when a CPT has repeating probabilities). We equate clauses and ADDs (more specifically, factors of the function ϕ from Algorithm 1) here

Table 3: Asymptotic upper bounds on the numbers of variables and clauses/ADDs for each encoding

Encoding(s)	Variables	Clauses/ADDs
`cd05`, `cd06`, `sbk05`	$\mathcal{O}(nv^{d+1})$	$\mathcal{O}(nv^{d+1})$
`d02`	$\mathcal{O}(nv^{d+1})$	$\mathcal{O}(ndv^{d+1})$
`db21`	$\mathcal{O}(nv)$	$\mathcal{O}(nv^2)$

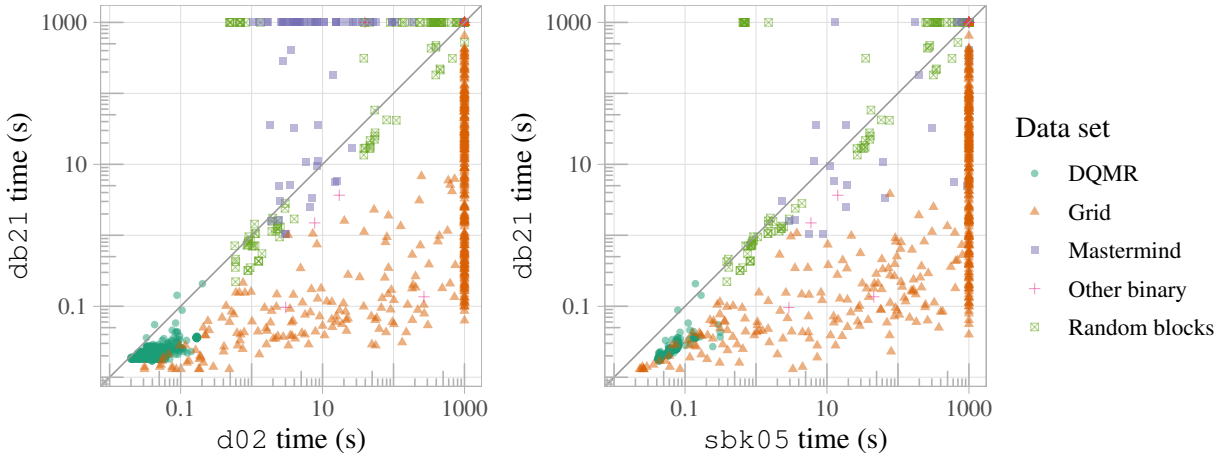


Figure 3: ADDMC inference time using `db21` compared to `d02` (left) and `sbk05` (right) for each instance in all data sets.

because ADDMC interprets each clause of any WMC encoding as a multiplicative factor of the ADD that represents the entire WMC instance (Dudek, Phan, and Vardi 2020). For literal-weight encodings, each weight is also a factor, but that has no effect on the asymptotic bounds in the table.

7 Conclusions and Future Work

- Bayesian networks and ADDMC are only particular examples. This should also work with Cachet (Sang et al. 2004).
  - Potential criticism may be that this doesn't allow us to use SAT-based techniques for probabilistic inference. However, they can still be used for a significant part of the encoding.
    - Zero-probability weights and one-probability weights can be interpreted as logical clauses. This doesn't affect ADDMC but could be useful for other solvers.
- Extra benefit: one does not need to come up with a way to turn some probability distribution to into a fully independent one.
- Important future work: replacing ADDs with AADDs (Sanner and McAllester 2005) is likely to bring performance benefits. Other extensions:
  - FOADDs can represent first order statements;
  - XADDs can replace WMI for continuous variables;
  - ADDs with intervals can do approximations.
- Filtering out ADDs that have nothing to do with the answer helps tremendously, but I'm purposefully not doing that.

A Proofs

Theorem 1. *The function μ_ν is a measure.*

Proof. Note that $\mu_\nu(\bot) = 0$ since there are no atoms below $\bot$. Let $a, b \in 2^{2^U}$ be such that $a \wedge b = \bot$. By elementary properties of Boolean algebras, all atoms below $a \vee b$ are either below a or below b. Moreover, none of them can be below both a and b because then they would have to be below $a \wedge b = \bot$. Thus

$$\begin{aligned}\mu_\nu(a \vee b) &= \sum_{\{u\} \le a \vee b} \nu(u) = \sum_{\{u\} \le a} \nu(u) + \sum_{\{u\} \le b} \nu(u) \\ &= \mu_\nu(a) + \mu_\nu(b)\end{aligned}$$

as required. □

Theorem 2. *For any set U and measure $\mu\colon 2^{2^U} \to \mathbb{R}_{\ge 0}$, there exists a set $V \supseteq U$, a factorable measure $\mu'\colon 2^{2^V} \to \mathbb{R}_{\ge 0}$, and a formula $f \in 2^{2^V}$ such that $\mu(x) = \mu'(x \wedge f)$ for all formulas $x \in 2^{2^U}$.*

Proof. Let $V = U \cup \{f_m \mid m \in 2^U\}$, and $f = \bigwedge_{m \in 2^U} \{m\} \leftrightarrow f_m$. We define weight function $\nu\colon 2^V \to \mathbb{R}_{\ge 0}$ as $\nu = \prod_{v \in V} \nu_v$, where $\nu_v(\{v\}) = \mu(\{m\})$ if $v = f_m$ for some $m \in 2^U$ and $\nu_v(x) = 1$ for all other $v \in V$ and $x \in 2^{\{v\}}$. Let $\mu'\colon 2^{2^V} \to \mathbb{R}_{\ge 0}$ be the measure induced by ν. It is enough to show that μ and $x \mapsto \mu'(x \wedge f)$ agree on the atoms in 2^{2^U}. For any $\{a\} \in 2^{2^U}$,

$$\begin{aligned}\mu'(\{a\} \wedge f) &= \sum_{\{x\} \le \{a\} \wedge f} \nu(x) = \nu(a \cup \{f_a\}) \\ &= \nu_{f_a}(\{f_a\}) = \mu(\{a\})\end{aligned}$$

as required. □

Lemma 1. *Let $X \in \mathcal{V}$ be a random variable with parents $\mathrm{pa}(X) = \{Y_1, \ldots, Y_n\}$. Then $\mathrm{CPT}_X\colon 2^{\mathcal{E}^*(X)} \to \mathbb{R}_{\ge 0}$ is such that for any $x \in \operatorname{im} X$ and $(y_1, \ldots, y_n) \in \prod_{i=1}^n \operatorname{im} Y_i$,*

$$\mathrm{CPT}_X(T) = \Pr(X = x \mid Y_1 = y_1, \ldots, Y_n = y_n),$$

where $T = \{\lambda_{X=x}\} \cup \{\lambda_{Y_i=y_i} \mid i = 1, \ldots, n\}$.

Proof. If X is binary, then CPT_X is a sum of $2\prod_{i=1}^n |\operatorname{im} Y_i|$ terms, one for each possible assignment of values to variables $X, Y_1, \ldots, Y_n$. Exactly one of these

terms is nonzero when applied to T, and it is equal to $\Pr(X = x \mid Y_1 = y_1, \ldots, Y_n = y_n)$ by definition.

If X is not binary, then $\left(\sum_{i=1}^{m} [\lambda_{X=x_i}]\right)(T) = 1$, and $\left(\prod_{i=1}^{m} \prod_{j=i+1}^{m} (\overline{[\lambda_{X=x_i}]} + \overline{[\lambda_{X=x_j}]})\right)(T) = 1$, so $\mathrm{CPT}_X(T) = \Pr(X = x \mid Y_1 = y_1, \ldots, Y_n = y_n)$ by a similar argument as before. □

Lemma 2. *Let $\mathcal{V} = \{X_1, \ldots, X_n\}$. Then*

$$\phi(T) = \begin{cases} \Pr(x_1, \ldots, x_n) & \text{if } T = \{\lambda_{X_i=x_i}\}_{i=1}^{n} \text{ for some } (x_i)_{i=1}^{n} \in \prod_{i=1}^{n} \operatorname{im} X_i \\ 0 & \text{otherwise,} \end{cases}$$

for all $T \in 2^U$.

Proof. If $T = \{\lambda_{X=v_X} \mid X \in \mathcal{V}\}$ for some $(v_X)_{X \in \mathcal{V}} \in \prod_{X \in \mathcal{V}} \operatorname{im} X$, then

$$\begin{aligned} \phi(T) &= \prod_{X \in \mathcal{V}} \Pr\left(X = v_X \,\middle|\, \bigwedge_{Y \in \mathrm{pa}(X)} Y = v_Y\right) \\ &= \Pr\left(\bigwedge_{X \in \mathcal{V}} X = v_X\right) \end{aligned}$$

by Lemma 1 and the definition of a Bayesian network. Otherwise there must be some non-binary random variable $X \in \mathcal{V}$ such that $|\mathcal{E}(X) \cap T| \neq 1$. If $\mathcal{E}(X) \cap T = \emptyset$, then $\left(\sum_{i=1}^{m} [\lambda_{X=x_i}]\right)(T) = 0$, and so $\mathrm{CPT}_X(T) = 0$, and $\phi(T) = 0$. If $|\mathcal{E}(X) \cap T| > 1$, then we must have two different values $x_1, x_2 \in \operatorname{im} X$ such that $\{\lambda_{X=x_1}, \lambda_{X=x_2}\} \subseteq T$ which means that $(\overline{[\lambda_{X=x_1}]} + \overline{[\lambda_{X=x_2}]})(T) = 0$, and so, again, $\mathrm{CPT}_X(T) = 0$, and $\phi(T) = 0$. □

Theorem 3. *For any $X \in \mathcal{V}$ and $x \in \operatorname{im} X$,*

$$(\exists_U(\phi \cdot [\lambda_{X=x}]))(\emptyset) = \Pr(X = x).$$

Proof. Let $\mathcal{V} = \{X, Y_1, \ldots, Y_n\}$. Then

$$\begin{aligned} (\exists_U(\phi \cdot [\lambda_{X=x}]))(\emptyset) &= \sum_{T \in 2^U} (\phi \cdot [\lambda_{X=x}])(T) \\ &= \sum_{\lambda_{X=x} \in T \in 2^U} \phi(T) \\ &= \sum_{\lambda_{X=x} \in T \in 2^U} \left(\prod_{Y \in \mathcal{V}} \mathrm{CPT}_Y\right)(T) \\ &= \sum_{(y_i)_{i=1}^{n} \in \prod_{i=1}^{n} \operatorname{im} Y_i} \Pr(x, y_1, \ldots, y_n) \\ &= \Pr(X = x) \end{aligned}$$

by the following arguments:

- the proof of Theorem 1 by Dudek, Phan, and Vardi (2020);
- if $\lambda_{X=x} \notin T \in 2^U$, then $(\phi \cdot [\lambda_{X=x}])(T) = \phi(T) \cdot [\lambda_{X=x}](T \cap \{\lambda_{X=x}\}) = \phi(T) \cdot 0 = 0$;
- Lemma 2;
- marginalisation of a probability distribution.

□

References

Bahar, R. I.; Frohm, E. A.; Gaona, C. M.; Hachtel, G. D.; Macii, E.; Pardo, A.; and Somenzi, F. 1997. Algebraic Decision Diagrams and Their Applications. *Formal Methods Syst. Des.* 10(2/3): 171–206. doi:10.1023/A:1008699807402. URL https://doi.org/10.1023/A:1008699807402.

Belle, V. 2017. Open-Universe Weighted Model Counting. In Singh, S. P.; and Markovitch, S., eds., *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, 3701–3708. AAAI Press. URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/15008.

Belle, V.; Passerini, A.; and Van den Broeck, G. 2015. Probabilistic Inference in Hybrid Domains by Weighted Model Integration. In (Yang and Wooldridge 2015), 2770–2776. URL http://ijcai.org/Abstract/15/392.

Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context-Specific Independence in Bayesian Networks. In Horvitz, E.; and Jensen, F. V., eds., *UAI '96: Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence, Reed College, Portland, Oregon, USA, August 1-4, 1996*, 115–123. Morgan Kaufmann. ISBN 1-55860-412-X. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1\&smnu=2\&article_id=359\&proceeding_id=12.

Chavira, M.; and Darwiche, A. 2005. Compiling Bayesian Networks with Local Structure. In (Kaelbling and Saffiotti 2005), 1306–1312. URL http://ijcai.org/Proceedings/05/Papers/0931.pdf.

Chavira, M.; and Darwiche, A. 2006. Encoding CNFs to Empower Component Analysis. In Biere, A.; and Gomes, C. P., eds., *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, volume 4121 of *Lecture Notes in Computer Science*, 61–74. Springer. ISBN 3-540-37206-7. doi:10.1007/11814948_9. URL https://doi.org/10.1007/11814948_9.

Chavira, M.; and Darwiche, A. 2007. Compiling Bayesian Networks Using Variable Elimination. In Veloso, M. M., ed., *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, 2443–2449. URL http://ijcai.org/Proceedings/07/Papers/393.pdf.

Chavira, M.; and Darwiche, A. 2008. On probabilistic inference by weighted model counting. *Artif. Intell.* 172(6-7): 772–799. doi:10.1016/j.artint.2007.11.002. URL https://doi.org/10.1016/j.artint.2007.11.002.

Chavira, M.; Darwiche, A.; and Jaeger, M. 2006. Compiling relational Bayesian networks for exact inference. *Int. J. Approx. Reason.* 42(1-2): 4–20. doi:10.1016/j.ijar.2005.10.001. URL https://doi.org/10.1016/j.ijar.2005.10.001.

Choi, A.; Kisa, D.; and Darwiche, A. 2013. Compiling Probabilistic Graphical Models Using Sentential Decision Diagrams. In van der Gaag, L. C., ed., *Symbolic and Quantitative Approaches to Reasoning with Uncertainty - 12th Eu-*

ropean Conference, ECSQARU 2013, Utrecht, The Netherlands, July 8-10, 2013. Proceedings, volume 7958 of *Lecture Notes in Computer Science*, 121–132. Springer. ISBN 978-3-642-39090-6. doi:10.1007/978-3-642-39091-3_11. URL https://doi.org/10.1007/978-3-642-39091-3_11.

Darwiche, A. 2001. On the Tractable Counting of Theory Models and its Application to Truth Maintenance and Belief Revision. *J. Appl. Non Class. Logics* 11(1-2): 11–34. doi:10.3166/jancl.11.11-34. URL https://doi.org/10.3166/jancl.11.11-34.

Darwiche, A. 2002. A Logical Approach to Factoring Belief Networks. In Fensel, D.; Giunchiglia, F.; McGuinness, D. L.; and Williams, M., eds., *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25, 2002*, 409–420. Morgan Kaufmann. ISBN 1-55860-554-1.

Darwiche, A. 2004. New Advances in Compiling CNF into Decomposable Negation Normal Form. In de Mántaras, R. L.; and Saitta, L., eds., *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, 328–332. IOS Press. ISBN 1-58603-452-9.

Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press. ISBN 978-0-521-88438-9. URL http://www.cambridge.org/uk/catalogue/catalogue.asp?isbn=9780521884389.

Dudek, J. M.; Phan, V.; and Vardi, M. Y. 2020. ADDMC: Weighted Model Counting with Algebraic Decision Diagrams. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 1468–1476. AAAI Press. ISBN 978-1-57735-823-7. URL https://aaai.org/ojs/index.php/AAAI/article/view/5505.

Fierens, D.; Van den Broeck, G.; Renkens, J.; Shterionov, D. S.; Gutmann, B.; Thon, I.; Janssens, G.; and De Raedt, L. 2015. Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory Pract. Log. Program.* 15(3): 358–401. doi:10.1017/S1471068414000076. URL https://doi.org/10.1017/S1471068414000076.

Gaifman, H. 1982. On local and non-local properties. In *Studies in Logic and the Foundations of Mathematics*, volume 107, 105–135. Elsevier.

Gogate, V.; and Domingos, P. M. 2016. Probabilistic theorem proving. *Commun. ACM* 59(7): 107–115. doi:10.1145/2936726. URL https://doi.org/10.1145/2936726.

Hoey, J.; St-Aubin, R.; Hu, A. J.; and Boutilier, C. 1999. SPUDD: Stochastic Planning using Decision Diagrams. In Laskey, K. B.; and Prade, H., eds., *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*, 279–288. Morgan Kaufmann. ISBN 1-55860-614-9. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1\&smnu=2\&article_id=178\&proceeding_id=15.

Holtzen, S.; Van den Broeck, G.; and Millstein, T. D. 2020. Dice: Compiling Discrete Probabilistic Programs for Scalable Inference. *CoRR* abs/2005.09089. URL https://arxiv.org/abs/2005.09089.

Kaelbling, L. P.; and Saffiotti, A., eds. 2005. *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*. Professional Book Center. ISBN 0938075934. URL http://ijcai.org/proceedings/2005.

Kimmig, A.; Van den Broeck, G.; and De Raedt, L. 2017. Algebraic model counting. *J. Appl. Log.* 22: 46–62. doi:10.1016/j.jal.2016.11.031. URL https://doi.org/10.1016/j.jal.2016.11.031.

Lagniez, J.; and Marquis, P. 2017. An Improved Decision-DNNF Compiler. In Sierra, C., ed., *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 667–673. ijcai.org. ISBN 978-0-9992411-0-3. doi:10.24963/ijcai.2017/93. URL https://doi.org/10.24963/ijcai.2017/93.

Oztok, U.; and Darwiche, A. 2015. A Top-Down Compiler for Sentential Decision Diagrams. In (Yang and Wooldridge 2015), 3141–3148. URL http://ijcai.org/Abstract/15/443.

Sang, T.; Bacchus, F.; Beame, P.; Kautz, H. A.; and Pitassi, T. 2004. Combining Component Caching and Clause Learning for Effective Model Counting. In *SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings*. URL http://www.satisfiability.org/SAT04/programme/21.pdf.

Sang, T.; Beame, P.; and Kautz, H. A. 2005. Performing Bayesian Inference by Weighted Model Counting. In Veloso, M. M.; and Kambhampati, S., eds., *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, 475–482. AAAI Press / The MIT Press. ISBN 1-57735-236-X. URL http://www.aaai.org/Library/AAAI/2005/aaai05-075.php.

Sanner, S.; and McAllester, D. A. 2005. Affine Algebraic Decision Diagrams (AADDs) and their Application to Structured Probabilistic Inference. In (Kaelbling and Saffiotti 2005), 1384–1390. URL http://ijcai.org/Proceedings/05/Papers/1439.pdf.

Somenzi, F. 2015. CUDD: CU decision diagram package release 3.0.0. *University of Colorado at Boulder* .

Van den Broeck, G.; Taghipour, N.; Meert, W.; Davis, J.; and De Raedt, L. 2011. Lifted Probabilistic Inference by First-Order Knowledge Compilation. In Walsh, T., ed., *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 2178–2185. IJCAI/AAAI. ISBN 978-1-57735-516-8. doi:10.5591/978-1-57735-516-

8/IJCAI11-363. URL https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-363.

Yang, Q.; and Wooldridge, M. J., eds. 2015. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. AAAI Press. ISBN 978-1-57735-738-4. URL http://ijcai.org/proceedings/2015.

Zhao, H.; Melibari, M.; and Poupart, P. 2015. On the Relationship between Sum-Product Networks and Bayesian Networks. In Bach, F. R.; and Blei, D. M., eds., *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, 116–124. JMLR.org. URL http://proceedings.mlr.press/v37/zhaoc15.html.